**Feature Development Plan Template**

**Feature Name**

*Role Based Access Control*

# Requirement Analysis

**User Story Reference:** *[Link or reference to the User Story] (optinal)*

**Summary:** *The RBAC implementation aims to enhance system security and access control by assigning specific roles to users, allowing or restricting their access based on predefined permissions. This feature is essential for providing a granular level of control over functionalities within the system.*

# Impact Analysis (Optional)

**Affected APIs:**

* *Authentication and Authorization APIs*
* *CRUD APIs for relevant sections*
* *Other role management APIs*

**Analysis:** *Introducing RBAC requires modifications to authentication and authorization mechanisms, as well as adjustments to existing CRUD APIs. Dependencies on user roles must be considered, and potential conflicts or areas requiring special attention should be addressed.*

# Database Schema Changes (Optional)

**Current Schema Overview:** *User collection: Contains user information, including username, password, email.*

**Proposed Changes:** *Enhance the user collection to include a 'role' field indicating the user's assigned role.*

# API Endpoints Design

**Endpoint Summary:**

* ***/api/Auth*** *(Authentication and Authorization)*
* ***/api/Project*** *(CRUD operations for relevant sections)*
* ***/api/RoleManagement*** *(Role management APIs)*

**Design Details:**

***/api/Project***

* *CRUD endpoints for each section with role-based middleware for authorization.*

***/api/RoleManagement***

* ***/assignRole****: POST request to assign a role to a user.*
* ***/removeRole****: POST request to remove a role from a user.*
* ***/getRoles****: GET request to retrieve available roles.*

# Pseudo Code for Key Functionalities

**Function Overview: *CheckUserPermission****(userId, requiredPermission).*

Function to check if the user has the required permission based on their role

**Pseudo Code:** *For each key function, provide pseudo code to illustrate the core logic and flow.*

Pseudo code is a way to describe the functionality of a program or algorithm in a format that’s more readable than actual code, but not as detailed as plain language. It helps in planning and understanding the logic before diving into the actual coding.

Let’s say the feature in question involves adding a new functionality to a user profile system, where the backend needs to handle a request to update a user’s email address. Here’s an example of how the pseudo code for this functionality might look:

**Pseudo Code for Update Email Function Functionality: *CheckUserPermission****(userId,requiredPermisson)*

**Name: CheckUserPermission** **Parameters:**

* ***userId****: User ID of the logged-in user.*
* ***requiredPermission****: The permission required for the operation.*

# Function to Check User Permission

FUNCTION checkUserPermission(userId, requiredPermission)

# Find user in 'users' collection by user ID

user = usersCollection.findOne({'\_id': userId})

# Check if user exists

IF user is not null THEN

# Check if user's role has the required permission

IF requiredPermission in user.role.permissions THEN

RETURN "User has the required permission"

ELSE

THROW "Unauthorized Access: User does not have the required permission"

END IF

ELSE

THROW "User not found"

END IF

END FUNCTION

This pseudo code outlines the basic logic for updating a user’s email. It includes input validation, checking if the user exists, and handling the database transaction, which are common steps in backend operations. Note that pseudo code is not meant to be executed, and the syntax is not language-specific. It serves as a blueprint for the actual code that will be written.

# Deployment and Configuration Changes (Optional)

**New App Settings:** *The default role assigned to a new user.*

**Deployment Changes:** *Configure RBAC settings during deployment.*

**Testing and Verification:**

* *Perform unit tests for each RBAC-related function.*
* *Conduct integration tests to ensure proper interaction with CRUD APIs.*
* *Manual testing for user registration, login, and role-based functionality.*